Access Control Lists

ACL (Access Control List) is a filtering mechanism at the L3, partly L4 layer. Standard ACL filters by IP addresses, extended ACL can also filter by ports and protocols.

Each ACL is identified by its number (numbered ACL) or name (named ACL).

Dedicated ACL number ranges for numbered ACLs:

1–99 or 1300–1399	standard ACL
100–199 or 2000–2699	extended ACL
remaining	AppleTalk, IPX, extended IPX,… (not used)

So I can create a standard ACL specified by number or name, and an extended one as well.

We create the ACL first, and then deploy it on an interface and direction (i.e., we filter on a specific interface, in the inbound or outbound direction). Also, a special type of ACL is an ACL to protect VTY lines.

An ACL is a sequence of rules (also numbered) – Access Control Entries (ACE), and a packet on a given interface and in a given direction must pass through this sequence. The packet either matches or does not match what the rule specifies, and if it does, then the rule determines what happens to the packet:

- if the packet satisfies a certain property (for example, the IP address belongs to a certain network), it can continue on its way (permit),
- if the packet meets a certain property, it should be discarded (deny).

The following situations can occur:

- the packet does not match the condition in the rule, so it goes to the next rule in the ACL,
- the packet matches the condition, it is either sent over the interface (permit) or discarded (deny), and it does not continue in the ACL,
- the packet goes through all the rules in the ACL sequentially and none of them match the condition (so the first bullet in this list always occurs), then it depends, at the end we usually have an implicit deny any rule, so it will be dropped.

ACL rules usually contain IP addresses to which we need to add a mask. Cisco uses wildcard masks for ACLs, i.e. inverse masks. A wildcard mask is created by inverting the 0 and 1 in a regular (binary) mask. A wildcard mask doesn't necessarily have to have only zeros at the beginning and only ones at the end, we simply specify that in an IP address at a given binary position either the digit prescribed by that address must be present, or conversely it doesn't matter if the packet at that IP address has a 0 or a 1 at that position.

If I want to put a specific unicast IPv4 address (one, no network) in an ACL rule, the regular mask would be 255.255.255.255, but the wildcard mask would be 0.0.0.0. Example:

```
172.16.0.12 0.0.0.0
```

If I want to specify this subnet: 172.16.0.0 255.255.0.0, it is:

```
172.16.0.0 0.0.255.255
```

If in the 10.0.0.0/8 network I want to capture in the ACL all packets with an IP address with a fixed number in the second octet:

```
10.96.0.0 0.0.255.255
```

Alternatively, only some bits of the octet are fixed:

```
10.96.0.0 0.15.255.255
```

The number 15 is binary 0000 1111, after inverting we get 1111 0000, so the upper half of the octet must be according to the formula. The number 96 is binary 0110 0000, which means that the upper half of the second octet must be 0110, the other half can be any value. For which of the following addresses would such a rule in an ACL be applied?

```
10.96.52.4
10.105.0.4
10.128.22.1
10.112.150.32
```

What if we mix it up? The rule would specify, for example, this:

```
10.96.32.0 0.0.15.223
```

What does that mean? Will the address 10.105.32.8 correspond to this regulation?

For a unicast address, we can specify the word host instead of the wildcard mask 0.0.0.0.

A standard numbered ACL is created as follows:

```
access-list 10 permit 10.96.0.0 0.0.255.255  ; source IP with the wildcard mask
access-list 10 deny host 10.96.100.84  ; host instead of the wildcard mask 0.0.0.0
access-list 10 deny any                ; not necessary, but useful for logging purposes
access-list 10 permit any              ; if we want everything else to go through
```

Possible actions:

permit, deny, remark

An ACL contains one or more ACEs, each with its own number.

To create a named ACL with two ACE entries (traditionally, the name is being capitalized):

```
ip access-list standard NO_ACCESS
      deny host 10.96.100.84
      permit 10.96.0.0 0.0.255.255
      exit
```

Deployment at the interface:

```
interface f0/0
      ip access-group 10 out          ; for outgoing traffic
      ip access-group 20 in           ; for incoming traffic
```

Verification:

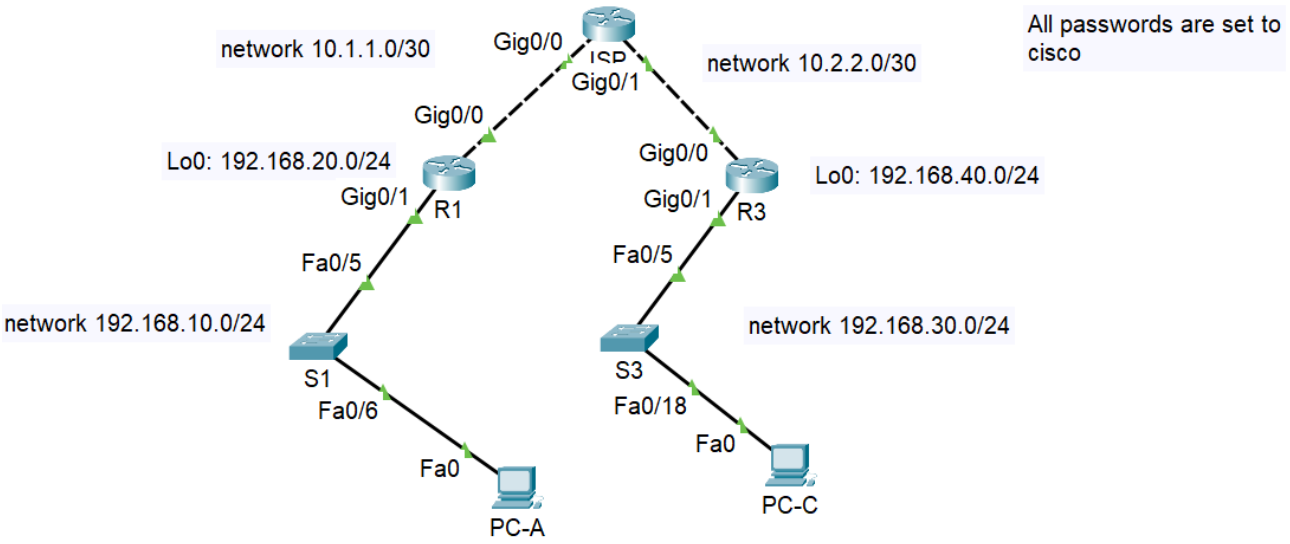
```
sh access-lists
sh ip access-list
sh access-list 10
sh ip int f0/0
sh run
```

Task:

On the web, there is a pre-prepared PKT file for standard ACLs. Download it and open it. Review all interfaces (including loopbacks) and networks. Verify that there are really no ACLs created on the routers. Verify that routing is working (which routing protocol is used?). Verify that the computers are reachable to each other.

Create a standard numbered ACL (No. 1) that allows traffic from the 192.168.30.0/24 and 192.168.20.0/24 networks to the 192.168.30.0/24 network, and disallow everything else leading to this network. Deploy the ACL on the appropriate interface. Remember, always deploy a standard ACL as close to the target as possible.

Use the appropriate show commands to view the ACL that you created and how it is deployed on the interface.



Verify that the ACL actually works – for both enabled and disabled networks. You can also use extended ping.

Next, create a named standard ACL, ACL_CONFIG, which allows traffic from the 192.168.40.0/24 network to the 192.168.10.0/24 network, and also allows PC-C (192.168.30.3) access to this network. Deploy on the appropriate device and interface.

Again, review the ACL you created and also the location where it is created.

Verify that the ACL works on both enabled and disabled connections. Also view the statistics using show access-lists.

Modification of Existing Standard ACLs

There are two ways to modify an existing ACL:

- I can create the whole ACL in a text editor (copy it from the console, modify it), remove the original one (no access list..., no ip access-list...) and copy it during the re-creation, because it is possible to copy to the terminal emulator),
- I use the commands to add new ACEs, I need to know which ACE numbers are free in the list.

For a named ACL:

```
sh access-lists          ; I'll look into which numbers are available (free)
ip access-list standard název
       30 permit address inv-mask
       40 deny any
       end
```

The ACL remains deployed on the interface even after the ACL is deleted or modified, so we don't have to re-deploy it on the interface after modification.

Note that although the deny any entry is valid by default, when we place it directly in the list, we can view related statistics.

Task:

Make this modification: devices from the network 209.165.200.224/27 should have access to 192.168.10.0/24, and the `deny any` ACE should be added to the end.

Test if the added items work as they should.

ACL to protect VTY lines

ACLs can also be deployed on VTY lines. This determines from where access to device administration will work.

ACLs are created in the same way as above, but the deployment to the links is slightly different:

```
line vty 0 4        ; or line vty 0 15, according to the number of VTY lines
      access-class ACL-name-or-number in
```

Task:

Simply enable telnet access on the R3 router (yes, we wouldn't normally do this, it's just for testing purposes here). Make sure it works from PC-A.

On R3, create a numbered ACL with the number 2 to allow access from address 192.168.10.3, and then access from addresses in the range 192.168.10.4 to .7, using a suitable inverse mask for the entire range.

It can be as follows:

```
access-list 2 permit host 192.168.10.3
access-list 2 permit 192.168.10.4 0.0.0.3
access-list 2 deny any
```

Deploy this ACL on the VTY lines on the router.

Verify that the ACL you created is working properly. Change the PC-A address to something outside the range allowed in the ACL and test if it passes.

On R3, view the ACL statistics (in the show ip access-list command) and check the filtering result.

On R3, configure SSH (add a suitably named user with a password) and change VTY access to SSH only. Test again.

VTY can also be used to protect lists on individual interfaces. But then we would have to deploy an ACL on each interface to protect Telnet/SSH access.